EasyChair Preprint

№ 7848

Machine Trainable Software Models Towards a Cognitive Thinking AI with the Natural Language Processing Platform NLX

Felix Schaller

April 27, 2022

Machine trainable software models towards a cognitive thinking AI with the natural language processing platform NLX

Felix Schaller[1]

Abstract: Since the last decade, machine learning, especially with artificial neural networks, has triggered a new quantum leap in computer science. Despite the considerable achievements, these applications still lack a general purpose approach for artificial intelligence (AI). The main reason is the absence of the ability for cognitive reflection or self-awareness. They are mainly highly specialized trained patterns that can solve intricate problems but cannot describe themselves. I would like to contrast this with a new method of trainable software models that shall be capable for self-awareness. Implemented in the project Natural Language Platform NLX. With this project it shall be demonstrated that self-aware AI is key for human-like cognitive tasks. The hypothesis claims that to reach this goal machines require to describe its system context semantically by a formal model. Neuronal networks are good at specific tasks, but the trained patterns cannot derive a reasoning for the trained solution only that it satisfies its intended functionality but not why. Creating formal model instead of patterns has turned out, that the formal nature of natural language is the best to reach that goal of a self-aware AI. Certainly there are other AI's that do natural language processing with neuronal networks. But most of the models try to resolve the content with too rigid constraints and with little attention to the context. For this project context plays a key role to resolve the meaning of natural language. If the context is resolved correct it can be used for general purpose tasks resolving anything imaginable.

Keywords: Machine Learning; Software Models; AI; Cognitive thinking

1 Machine learning today

The last decade can be seen as the great breakthrough in artificial intelligence, where finally by the leading role of neuronal networks, computation solutions could be achieved which where not possible before by using primarily formal methods and algorithms. Meanwhile almost all groundbreaking improvements by AI go on the account of neuronal networks. They perform superior especially in domains where there are no formal solutions possible, because no formal rules are available. Such domain is in particular the computer vision domain where patterns play a major role. But also for other domains like natural language processing, neuronal networks meanwhile play a key role to solve these tasks. In the last few years the project openAI [Op21] with their newest language model GPT-3 [FC20] has achieved astonishing results, where this AI can even create introcate softwarecode from natural language [Za21] that is even executable. Also in other domains of natural language

[1] Validas AG, IT research, Arnulfstr 27, 80335 München, Germany schaller@validas.de

processing GPT-3 in conjunction with neural theorem prover has achieved astonishing results by solving some math Olympics problems [Po22]. But yet when summarizing all these success stories the impression prevails, that these solutions can be good at a specific domain, but yet still require a human guided setup or operation frame to master the task. In other words it needs to be specialized for the domain to perform. Meaning that it would not be capable to develop the solution by it's own neither to reflect it's result. Something you would expect by a general purpose tool. So it can be confidently stated it is yet a stupid - indeed a very efficient stupid - machine.

2 The nature of neuronal networks

What makes us confident in that claim? the reason lies in the nature of neuronal networks itself. Though if stacked together in huge Networks they can even easily surpass a human on that specific domain, but still they can only perform tasks for what they are designed for. It can be anyway seen as a general purpose tool from that perspective, that the problem posed on can be general purpose indeed. But yet it processes only task for which it is specifically tailored for and not any task by one single interface. The reason is this networks does the task, but it does not know why. It doesn't even know if the result produced is correct except that it satisfies an intended cross reference. But here lies the rub, that to develop unique solutions the AI requires a formal reason.

Neuronal networks are not capable by nature for that task because they currently supersede on problems nobody knows a formal solution for, so you unleash a neuronal network on it to intuitively find a solution with heuristic methods, like humans develop a gut feeling for a certain domain by experience a network achieves positive success rate over time. After achieving a finite amount of training the network became confident in estimating the result. But no matter how precise the network performs, it's yet an estimation - indeed a highly sophisticated estimation - of the solution. This solution is anyway just a product of the training set posed to the network and in an unluckily constellation in the selection of training material during training, the network later may show unintended behavior leading to false positives [FS10]. One reason are unknown patterns on a sublimal level or patterns that seem not obviously relevant by human interpretation being mixed in the training data which influences the final result. Another example are adversarial attacks by noise. By nature a neuronal network is not so resistant against such type of signal jamming [NKP19]. Here the major reason is, that the network is applied to the raw data finding patterns on the atomic level. Would the signal instead be transposed by Fourier Transformation, a noise jamming could be isolated easily [ECK19].

2.1 Functional safety

Based on the fact, that no matter how sophisticated the network is, it is at the end still a highly sophisticated guessing, achieved through training experience by no formal rule. That makes it to the ultimate tool for unresolved problems but on the other hand makes those networks actually insufficient for functional safety domains [GB19]. Currently there are several safety standards under development which discuss criteria and evaluation methods under which circumstances they are acceptable [Ts20] [et20]

2.2 The solution is encrypted in the matrix

The final Problem then is, that the trained solution is finally hidden in some patterns of the various matrices inside the network, which does by no way unveil its reasoning for the solution. Ironically by the reason that the network was the mean of choice because there was no formal method available. In this problematic there lies a huge potential towards a paradigm shift about the meaning of formal methods and machine learning. This paper also addresses the current paradigm of formal truth in later chapters.

2.3 Natural language processing and neuronal networks

As natural language has yet been seen as too ambiguous for formal methods the solution lays near to use that available tool that can handle problems with unknown formal rules. And as the success story of the GPT-3 model reveals, it does this job pretty well and precise although language is seen as something too ambiguous for deterministic tasks like creating software code. The reason is, that the hypothesis of the ambiguous nature of language is actually not true. It is only true if you look on literal terms as formal expressions. And that's why solving natural language was yet prevailed to the domain of neuronal networks. But Language has indeed a very formal nature and can be very powerful if you look at natural language from a different point of view. This matter will be very essential for the topic of this paper and will be dedicated in particular in a later chapter.

3 The formal method and machine learning

As shown in the chapters before it is obvious that machine learning methods in particular with neuronal networks computation has penetrated into areas which were not available before with formal methods. But as shown yet it does not provide a reasoning, at least and most for itself to build formal conclusions from it. The simple reason is that to know what I am dealing with i require to describe it semantically furthermore this semantics requires a context. The context becomes key in that matter, but more of it later...

3.1 Structural science and the Hilbert program

When looking today on the domains of math and information science it is all about reasoning. In particular universal validity of claims and their proof. This was believed till Alan Touring introduced its Touring Machine as answer to David Hilbert's Program for an axiomatic proof theory [Pl14]. Touring demonstrated with this machine theory, that there is no ultimate law, that a task can be solved. This paradigm thus defines the fundament of all information theory, but it is only half true. It is only true when applied in the frame and to problems of the David Hilbert Program. And the problems stated there are the backbone of all formal problems in information technology till today. Basically what all axiomatic theorems do, is to set up a claim that pretend to have an ultimate validity and the algorithm has to be just sophisticated enough to handle that problem. If it does not handle it it means it has bugs and have to be fixed. But the other truth is, that the constructed reality of the designer of the axiom has put his imagination of reality into that formal rule. With the result, that designer figures out that not the algorithm is wrong but his imagination of the constructed reality does. Thus becoming a software developer is becoming humble about the validity of constructed realities. But that's not the point here, but shows an interesting side effect when developing algorithms and theorems. The real point for all formal problems is that they fail

1. because they claim to have ultimate validity. All constructed models that reflect a certain aspect of a real problem are true only in a certain area. No model can claim from itself being valid for all circumstances.
2. if a Touring Machine fails because it does not return from its pushdown state it means the problem is underspecified and lacks constraints it requires to solve the posed problem.
3. because of the self-entitlement of ultimate validity of axiomatic theorems they claim to work context free.
4. due the negligence of meta structures in the data those axioms are applied on raw and unstructured data on an atomic level. Thus lacking a structured semantic of the data any formal method will fail.

3.2 The key role of context

Soon or later everyone will come to the point where the limits of an axiomatic theorem or formal claim are shown up. Try to qualify a software for a functional safety domain. Here it is mandatory to draw a line around finite tasks, named use case, of within they can be declared as safe after undergoing a certain process of validation. Any contemporary program is designed to work in any context and the degree of freedom in that program is reduced to that amount of state to work safe under under any circumstance. Try to develop a Grammar for a domain specific language that can be unleashed on any text to resolve the

document structure from it in a document object model. It will work on formal languages, that's what they where designed for, but there will be no ultimate valid grammar to solve the content without a certain knowledge of the context of the text. Or try to automate tasks like continuous integration it requires always someone who sets it up manually. Thus without knowledge or consideration of the dealt context any theorem will fail.

3.3 Fractals and meta structures of reality

The reason why formal methods fail is because the reality has a self similar nature. Not that it is just self similar, those self similar features build upon meta structures which build the ground truth and are the backbone of any cybernetic system. Gregory Bateson a pioneer in cybernetics and systems theory shaped the term of "meta structuresïn his book Mind and Nature [Ba79]. He sees those structures as immanent and as a implicit consequence of growing complexity in systems. This means for a practical application, that without a proper knowledge of the context you are unable to distinguish self similar features from each other on an atomic level. Many different problems are the same and thus cannot be distinguished, which leads to an ambiguous situation how to categorize the data features. Unfortunately most applications work solely on the atomic level. Simple experiments with the human perception can be made to understand the need to create meta structures of the given data and relate them to the given context. Thus it seems obvious that for a proper

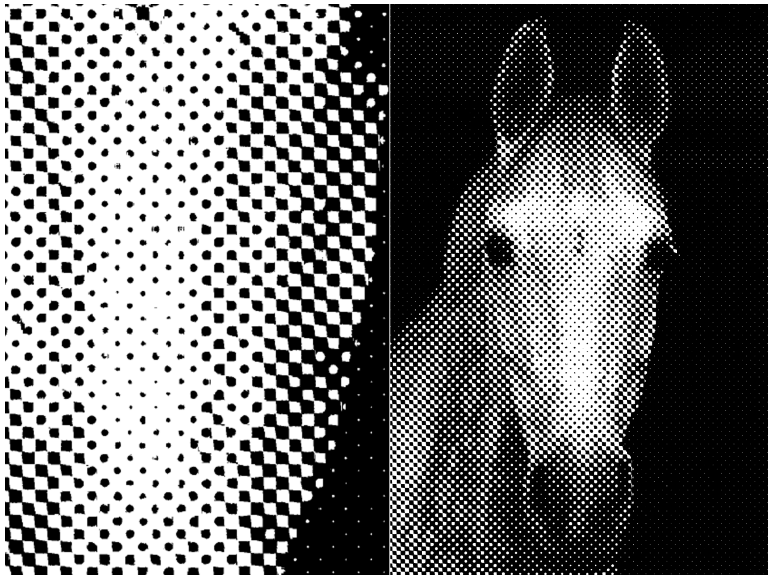

Fig. 1: (left) a closeup view on the raster does not reveal the motive, while on the right picture the motive can be clearly percepted

interpretation of posed problems it will be essential to develop a structured semantic for

them that they can be formally resolved. Otherwise the problem gets stuck in ambiguity. So the next question arises here: how to build up meta structures on data to derive a context so it can be processed with formal methods.

4 Towards formal models in AI

Carrying on with the success story of neuronal networks from above I mentioned the circumstance that they penetrate in areas where there is no formal rule known by applying heuristic methods of intense training. By training the neuronal network derives hidden meta structures of the trained content and stores it in its patterns of several hidden layers. But those meta structures do not own a semantic. So they can not explain what they are doing and develop reasoning for that. Thus it requires to develop a semantic to interpret those meta structures.

4.1 Introducing Software Models to reduce complexity

Even when working with software models which pretend to reduce complexity, like models from the SysML and UML domain they can help to sustain an overview for the user and if done by experienced System Engineers they can speed up the development process and can in some cases even generate software code. But as the modeling languages UML or SysML want to reserve the flexibility to describe systems, users can create huge mess. Many modeling automation projects in the industry, starting with great ambitions aiming to automate the development process and to leverage systems and software development to the next level not seldom died an inglorious death. Some of them are still riding a dead horse because the companies already poured millions of their budged into the project fearing all that investment to be lost if they redesign their concept from the bottom up, so they insisted to carry on and exacerbate. I have consulted some of them and was not reluctant to provide them with an honest analysis of the tricky situation. The main reason in those projects were that there was no proper Modeling semantic given that maintained a certain methodology hygiene. Especially because in many modeling tools available on the market there is neither type checking nor other methodology checking automatism implemented with the final result that they are useless for post-processing in downstream system domains and can finally only serve as a form of a documentation of the process. This circumstance arises the need for a proper modeling theory. For that purpose the FOCUS theory of distributed systems by Broy et.al. was developed [BS01] This modeling theory is currently implemented in the SPES modeling framework which is constantly being developed further in various research project and industrial applications [Bö16]. Having a proper concept of various viewpoints and the concept of granularity levels, the framework does a good job in improving the Quality of model based systems. Meanwhile it has been widely established in the industry and added with a proper tooling it has the potential becoming the new standard in systems engineering.

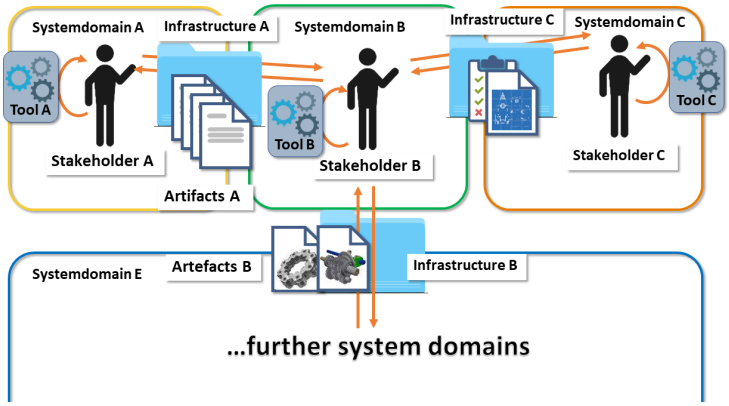

Fig. 2: Exchanging data artifacts across system domains require intense manual administration

4.2 Getting the developers knowledge into the model

As shown in the last chapter that a proper methodology accompanied with a powerful framework is key for a good success story in model based software engineering, and if such a proper methodology is properly implemented it can avoid nightmares in systems engineering. But although the model based approach improved the development in software and systems those models still requires the interpretation of the developer because those models do not own its own semantics. To the computer all software models are simply a finite collection of nodes and edges - nothing more. Yet that is nothing that could live up with the fancy claim of a cognitive thinking AI as promised in the title, but we are not there yet. This is the case simply due to the lack of knowledge for the machine - and that remains with all data artifacts in information technology. They create meaning for the developer or expert but not to the machine, simply because they hold no meta information for the machine about what they are and how they have to be treated. First and most, literals that are used by the developer to identify and classify model elements give meaning to the developer to understand what it is about, but the machine lacks an interpretation of the literals to create its own semantic.

4.3 Introduction of Meta Information

A first step would be to implement a semantic by introduce meta information as an overlaying layer that tells the machine how to handle model elements. this meta information introduce a classification system with a proper taxonomy to describe a model element lets say we have a model element that shall represent a specific tool e.g. a GCC compiler then we assign a class hierarchy which derives that specific compiler tool from a generic compiler

up to higher super classes tool, executable etc. further it is known that that tools has certain interfaces. Now when we assign that class to a tool in a tool chain the modeling tool knows what kind of data that tool can interchange with other tools. If we now want to define a artifact flow from one tool to another, the modeling tool already can do some automatic assignment because by the typing of the tools it can derive constraints of what kind of data can be interchanged between those tools. and the modeler does no longer need to explicitly assign the data connections by himself but the software already knows what connections are possible and can connects them by themselves. Here at that stage it would be already a semi automatic and proactive tool. Such proactive actions can be a Help, but not seldom they can also be very annoying.

4.4 Interactive modeling and machine trained models

Many professional computer users who where already using Microsoft computers around 1995 are very likely to have come in touch with Microsoft®Word™back then. At that time Word had an digital assistant who tried to guess the users current actions. So if you were typing a postal address the digital assistant appeared from nowhere saying “Ah obviously you want to write a letter”. In some cases the guess would have been right. But honestly in most cases this little lad was more than an annoying nightmare, because in most cases the unordered and spontaneous interventions in form of an interactive assistant where inappropriate for the current situation. The main reason was, that the solutions offered where too stenciled that it could have been addressing real individual needs for the user. This originated from the cause that you could not interact with that assistant like with a real human person and respond: “sorry you misinterpreted, this is not going to become a letter but an address list”. Then next time the assistant would have behave different. But because it just imitated basic tasks based on a stencil this would have not lead to a real semantic to source the specific needs of the user. The Clippy assistant from Microsoft®Word™had a precessor: Microsoft®Bob that originated from an research project Seductive Interfaces by Clifford Nass and Byron Reeves [Sk94]. Based on the research of what can improve Human Computer interaction those stencils for the assistant are created. But human computer interaction cannot satisfy stencils. It requires a real interactions with learning effect. So this means that if the computer want to response to the need of the user they require a model that can be trained. How such models can be trained? following approaches are possible:

- macro training by recording tasks in a guided way. so the computer learns by observing user interactions with a GUI and then repeats it on similar problems. Through an interactive approach in the case of conflicts the model can further improve to learn to handle special cases
- Back projection of error. By reverse analyzing logged action the user gives feedback about what was made right and what wrong thereby proactive processes can interactively adapt on the user’s need.

Although this can improve the annoying behavior of proactive assistant this already opens up an interface for supervised learning based on software models that can be trained. the interface has much in common with declarative programming language but with interactive additions to improve by failure. The interactive approach bypasses the problem that programming implements the constructed worldview of the developer into the solution, which is often incorrect, while an interactive approach is validated by reality itself. And last but not least many conflicts arise by a different interpretation of a topic between different parties. Through own studies with complex software tools like professional 3D programs i made the experience that supervised training of models to speed up redundant workflows works already on the level of GUI interaction but still comes on the cost of manual work because pure interaction lack a fully scalable semantic [fr12]. To improve the training it would require an interface which would enable the interaction via natural language

4.5 A theory about natural language and its origin

When discussing the topic of artificial intelligence the performance of an artificial intelligent systems are usually tried to be benchmarked to human intelligence and how far this system would be away from a real cognitive thinking and human-alike system. Yet many artificial intelligent systems surpass humans on a certain discipline with ease but they could be anyway compared rather to an animal stage. the main reason is not so much the power of the artificial system itself but the ability to self reflect and analyze. Human consciousness is superior to the animal not solely of the bigger brain but because they developed a tool of cognitive perception of the environment and itself. This tool is called language. Only by the tool providing semantic to a given context the cognitive achievement for the human kind was possible. And this ability is the only thing that makes them superior to the animal.

So How could language have been evolved? Studies with primates and kids in the field of cooperability showed that human children intuitively cooperate while primates rather compete against each other. A primate knows that the stronger partner does not share the reward with the other while kids instinctively shared the reward [HT06]. This arises the thought that the success of the human race might lie in the ability to cooperate and form enterprises to accomplish tasks to what a single individual would not be capable to. For example hunt a mammoth or buffaloes. For such a venture it requires to synchronize the strategy and to coordinate the tasks among the members of the venture. therefore a tool is required to transfer the vision and strategy of the leader to all the members. In the same context the cave paintings may also have been arised to support the verbal expressions.

4.6 language as a model

Talking about natural language processing in the previous chapters, the current practiced approach is achieved by neuronal networks. Language expressions are yet widely understood

as too ambiguous to formalize expressions and theorems. But this is only half true. Language is a tool that can be very precise in expressing. Looking at language as formal commands doesn't work. Because language explicitly requires a context to work. By its declarative nature, semantic in language is created by interaction with the other party. this is important because

- terms in language are not of axiomatic nature and thus do not have a fixed meaning. So the real semantic of a term is either constrained by the context or requires to be negotiated with the other party till both parties reached an consensus.
- it has a fractal nature and that makes language a very efficient modeling tool where it only requires detailed descriptions where it does not derive automatically by the given context
- a further fractal feature of language is a proper taxonomy of the terms where terms derive from a hierarchic taxonomy

This and more features of natural language makes it very efficient in the expression and communication of intentions because many things can be implicitly derived from the context and do not have to be explained in a redundant manner. Through its truly declarative nature, language language has the ability to build formal models that can be used for cognitive human like problems and tasks.

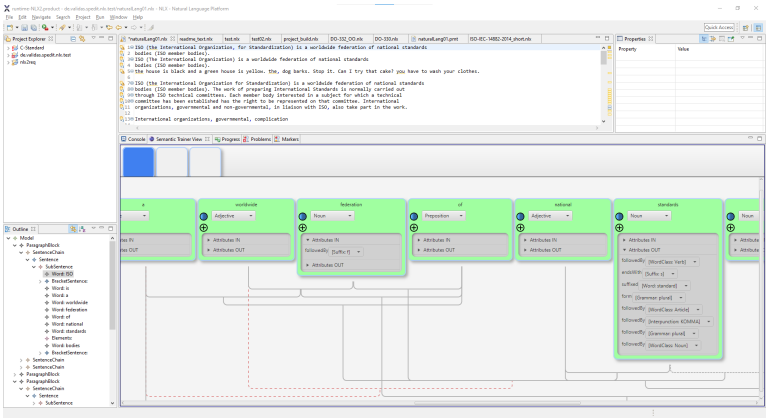

Fig. 3: Overview of the NLX tool. (top) text document, (bottom) interactive grammar trainer, (bottom left) DOM-tree

5 The natural language project NLX

Based on the prior considerations about natural language processing in a truly formal manner with machine trainable software models, the NLX project was set up. Currently

aimed on the goal to provide a proof of concept for the hypothesis of natural language based and machine trained models. Also to show that such models are capable to resolve cognitive tasks and provide a argued reasoning for its conclusions. Something that is not yet possible with alternative solutions available in the field of artificial intelligence. The project is built up in Java with the Eclipse modeling framework and the help of Neo4j as a spatial graph database to formalize the semantic structure and to resolve patterns with the graphical querying language Cypher.

As a base platform it uses a DSL grammar to resolve natural language into a document object model tree (DOM-tree). On top of this tree all kinds of generators can be adapted to generate other XMI-Models from the structure, but also do natural language processing. Currently the development of the natural language processing is in the state of developing a grammar trainer which resolves the sentences in a grammar model where in the future an interlinked ontology model and constraint proovers will be built on top. Those proovers then would have the role to validate the created context of the statements and derive formal processes from it like analysis tasks, process automation, code generation and many more

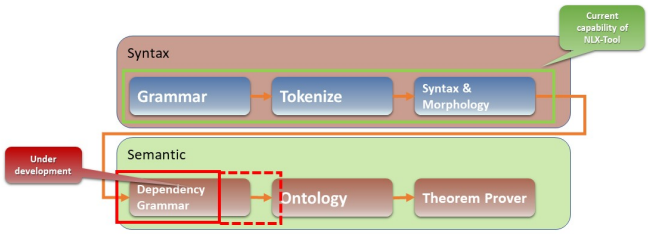

Fig. 4: Current development status of the project

5.1 grammar tree

Currently the project is in the state to finalize the grammar trainer. With this module it is intended to train a model that is capable to resolve a sentence structure and separates the parts of subject predicate and objects an further determine time and mode like passive, active, conjunctive. the resolved sentence structure shall then be transferred into an ontology model which defines the entities, the attributes and its functional relation to each other. this document-internal ontology is then linked to other related ontologies outside the document or with a database of ontologies acting as the background knowledge of the system. all this is then fed into a kind of sayconstraint proover which does the validation of the statements

on one hand and resolves the logic on the other hand. The grammar trainer already can be seen as a first proof of concept of machine trainable models. It attempts to resolve the sentence structure like finding an exit through a maze. if the entire sentence has found one root and all branches cover the entire sentence the sentence is resolved. the resolved sentence will then be transferred into the overall ontology of the document. The solution of a trainable grammar is chosen by the reason, that constructed rules would be too unflexible for the almost infinite variances of sentence structures, thus the structures shall be trained with a training interface to improve the grammar model constantly. Currently the extension of the training capabilities are ongoing and are estimated to be complete soon.

References

[Ba79] Bateson, G.: Mind and nature: A necessary unity. New York: EP Dutton. Bateson, MC (1994). Peripheral visions: Learning along the way, 1979.

[Bö16] Böhm, W.; Daun, M.; Koutsoumpas, V.; Vogelsang, A.; Weyer, T.: SPES XT Modeling Framework. In (Pohl, K.; Broy, M.; Daembkes, H.; Hönninger, H., eds.): Advanced Model-Based Engineering of Embedded Systems: Extensions of the SPES 2020 Methodology. Springer International Publishing, Cham, pp. 29–42, 2016, ISBN: 978-3-319-48003-9, URL: https://doi.org/10.1007/978-3-319-48003-9_3.

[BS01] Broy, M.; Stølen, K. In: Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement. Springer New York, New York, NY, 2001, ISBN: 978-1-4613-0091-5, URL: https://doi.org/10.1007/978-1-4613-0091-5_2.

[ECK19] Esmaeilpour, M.; Cardinal, P.; Koerich, A. L.: A robust approach for securing audio classification against adversarial attacks. IEEE Transactions on Information Forensics and Security 15/, pp. 2147–2159, 2019.

[et20] et.al., J. M. C.: Concepts of Design Assurance for neural networks (CoDANN) - easa.europa.eu, Mar. 2020, URL: https://www.easa.europa.eu/downloads/112151/en.

[FC20] Floridi, L.; Chiriatti, M.: GPT-3: Its nature, scope, limits, and consequences. Minds and Machines 30/4, pp. 681–694, 2020.

[fr12] freshNfunky: Freshnfunky/Intelligentmaya: Cross Breeding Artificial Intelligence with 3D application, 2012, URL: https://github.com/freshNfunky/intelligentMaya.

[FS10] Forman, G.; Scholz, M.: Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. Acm Sigkdd Explorations Newsletter 12/1, pp. 49–57, 2010.

[GB19] Gharib, M.; Bondavalli, A.: On the evaluation measures for machine learning algorithms for safety-critical systems. In: 2019 15th European Dependable Computing Conference (EDCC). IEEE, pp. 141–144, 2019.

[HT06] Herrmann, E.; Tomasello, M.: Apes' and children's understanding of cooperative and competitive motives in a communicative situation. Developmental Science 9/5, pp. 518–529, 2006.

[NKP19] Naseer, M.; Khan, S.; Porikli, F.: Local gradients smoothing: Defense against localized adversarial attacks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, pp. 1300–1307, 2019.

[Op21] OpenAI, June 2021, URL: https://openai.com/.

[Pl14] von Plato, J.: The development of proof theory, Oct. 2014, URL: https://plato.stanford.edu/entries/proof-theory-development/.

[Po22] Polu, S.; Han, J. M.; Zheng, K.; Baksys, M.; Babuschkin, I.; Sutskever, I.: Formal mathematics statement curriculum learning. arXiv preprint arXiv:2202.01344/, 2022.

[Sk94] Skelly, T.; Fries, K.; Linnett, B.; Nass, C.; Reeves, B.: Seductive interfaces: Satisfying a mass audience. In: Conference companion on Human factors in computing systems. Pp. 359–360, 1994.

[Ts20] Tsukiyama, A.: ISO/PAS 21448:2019, June 2020, URL: https://www.iso.org/standard/70939.html.

[Za21] Zaremba, W.: OpenAI Codex, Nov. 2021, URL: https://openai.com/blog/openai-codex/.